(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(72) Inventors: **BASSO, Claude**; 7604 Percy Court, Raleigh, NC 27613 (US). **VAIDHYANATHAN, Natarajan**; 3611-13K University Drive, Durham, NC 27707 (US). **WOODLAND, Gail, Irene**; 123 Booth Road, Chapel Hill, NC 27516 (US).
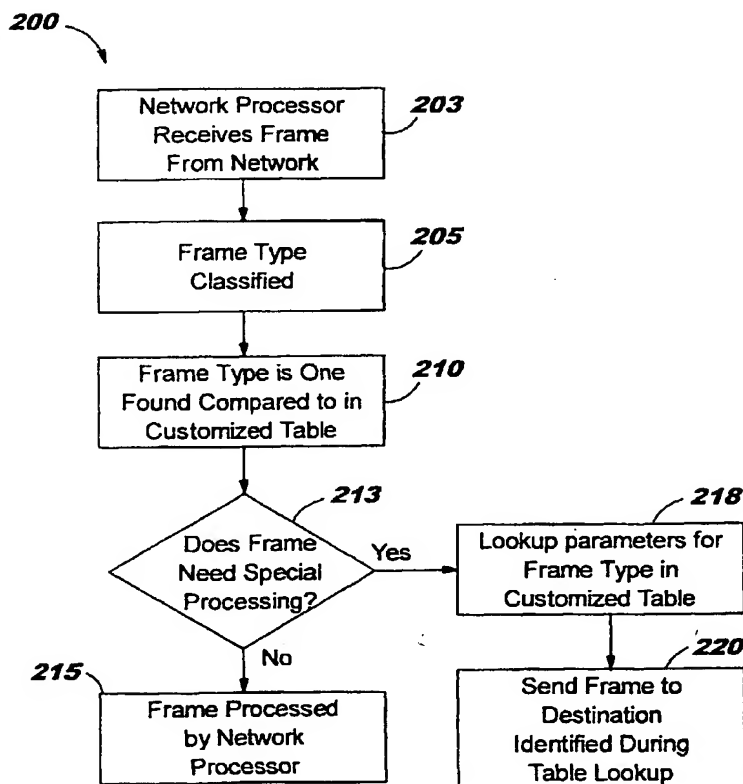
[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR CUSTOMIZING AND FORWARDING PARAMETERS IN A NETWORK PROCESSOR

(57) Abstract: In a distributed networking environment employing several general purpose processors (i.e., control point processors) for controlling one or more network processor devices, a mechanism for distributing processing across several general purpose processors and interface for configuring a network processor so that specific general purpose processors handle specific operations in a large networking environment, thus, reducing requirement for provisioning a plurality of protocol stacks on each general purpose processor.

200

Network Processor Receives Frame From Network — **203**

Frame Type Classified — **205**

Frame Type is One Found Compared to in Customized Table — **210**

Does Frame Need Special Processing? — **213**
— Yes → Lookup parameters for Frame Type in Customized Table — **218**
— No → Frame Processed by Network Processor — **215**

Send Frame to Destination Identified During Table Lookup — **220**

WO 01/77849 A2

patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Published:
— *without international search report and to be republished upon receipt of that report*

METHOD AND APPARATUS FOR CUSTOMIZING AND FORWARDING PARAMETERS IN A NETWORK PROCESSOR

BACKGROUND OF THE INVENTION
Field of the Invention

This invention relates generally to network processor devices, and more specifically, to a mechanism for distributing processing across several general purpose processors (i.e., control point processors).

Discussion of the Prior Art

Figure 1 depicts a typical networking configuration 10 including a single General Purpose Processor (GPP) control device 15 controlling many network processor devices 25a,..,25n. A processing bottleneck is often created in the system 10 at the GPP since the GPP handles all special data packet (frame) types (eg Point-to-Point (PPP) control frames or unknown frames).

Figure 2 illustrates a more distributed networking configuration implementing multiple General Purpose Processor control devices 15a,.., 15n. In this configuration, in order for a network processor to forward a special frame type to a GPP for processing, each GPP requires protocol stacks for each protocol it must handle since the network processors are not able to decipher how to send a special frame to a specific GPP. A requirement that each GPP have a complete protocol stack for each protocol handled by the Network Processor increases memory requirements and lowers performance.

Summary of the Invention

Accordingly, the invention provides a system for distributing processing of special data frames received by a network processor (NP) device in a distributed networking environment comprising one or more general purpose control processors (GPP) that control at least one NP devices, said system comprising a configurable table implemented in said at least one NP device for mapping special data frame types received at said NP device with a target address and a target port address associated with a target GPP capable of handling said special frame type, wherein said NP network processor includes mechanism for classifying the received data frames and comparing each said frame to entries in said configurable table, and said network processing device includes means responsive to

said classified frame type having a corresponding entry in said table, for forwarding said frame to said target GPP through a corresponding target port address, wherein said target GPP is provisioned for handling forwarding of said special data frame type.

According to a preferred embodiment a GPP configuration in a distributed networking environment that obviates the need for provisioning a complete set of protocol stacks for each GPP device is provided.

Preferably a mechanism for configuring GPPs in a distributed networking environment to handle specific special frame types and to provide the capability of distributing special frame types to a specific, configured GPP by a network processor is provided.

Further an Application Program Interface (API) and method for distributing processing across several general purpose processors (i.e., control point processors) and enabling a network processor to configure which processor handles specific operations is provided.

In a further aspect, the invention provides a method for distributing processing of special data frames received by a network processor (NP) device in a distributed networking environment comprising one or more general purpose control processors (GPP) that control one or more NP devices, said method comprising:

a) providing a table implemented in said NP device for mapping special data frame types received at said NP device with a target address and a target port address associated with a target GPP capable of processing said special frame type;

b) receiving at said NP network processor a data frame of a data frame type;

c) classifying the received data frames and comparing each said frame to entries in said table; and

d) if said classified frame type has a corresponding entry in said table, forwarding said special data frame to said target GPP through a corresponding target port address, said target GPP provisioned for handling forwarding of said special data frame type.

Advantageously, such a method preferably enables a network processor to distribute processing across several general purpose processors (i.e., control point processors) and enables the capability to configure which processor handles specific operations in a large networking environment thus reducing the number of protocol stacks that need to be maintained on each general purpose control processor.

In a yet further aspect the invention provides an application programming interface for configuring a network processor (NP) device operating in a distributed networking environment including one or more general purpose control processors (GPP) that control one or more NP devices, said interface comprising: a mechanism for generating a data structure for input to said NP device, said data structure having one or more entries specifying a special frame type and associated target address and target port address associated with a GPP in said networking environment capable of handling said special frame type; and a device for generating a customized table from said generated data structure, and inputting said customized table to an NP device memory, whereby subsequent table lookups in said table are performed for forwarding special frame types to an associated GPP through a target port address.

In a still further aspect the invention provides a program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for configuring a network processor (NP) device operating in a distributed networking environment including one or more general purpose control processors (GPP) that control one or more NP devices, said method steps comprising:

(a) generating a data structure for input to said NP device, said data structure having one or more entries specifying a special frame type and associated target address and target port address associated with a GPP associated with said networking environment capable of handling said special frame type; and,

(b) generating a customized table from said generated data structure, and inputting said customized table to NP device memory, whereby subsequent table lookups in said table are performed for forwarding special frame types to an associated GPP through a target port address.

Preferably, the data structure includes a string of tagged list values, each tagged list value including an attribute field representing a

special frame type to be included in said table, a length field
representing a length of said tagged list value, and, a value field for
specifying a target associated target address and target port address for
the special frame type.

Brief Description of the Drawings

A preferred embodiment of the present invention will now be
described, by way of example only, and with reference to the following
drawings:

Figure 1 is a general block diagram of a distributed network
processing environment including a plurality of network processing devices
controlled by a single general purpose processor (GPP) device.

Figure 2 is a general block diagram of a distributed network
processing environment including a plurality of network processing devices
controlled by multiple general purpose processor (GPP) devices.

Figure 3 illustrates an example customizable table 100 for
forwarding the special frame types according to a preferred embodiment of
the present invention.

Figure 4 is a flow chart showing how processing is distributed
across several general purpose processors according to a preferred
embodiment of the present invention.

Detailed Description of the Preferred Embodiment

According to a preferred embodiment, an application program
interface (API) is provided which is implemented by a General Purpose
Processor (GPP) control device operating in a distributed networking
environment (e.g., external GPP, embedded GPP in network processor, etc.)
for enabling the customization and configuration of network processor (NP)
devices so that they may handle the forwarding of special data packets
(frame) types. For exemplary purposes, reference is made to one
networking processor-based device and system such as described in
commonly- owned, co-pending U.S. Patent Application Serial No. 09/384,691
filed August 27, 1999 and entitled "NETWORK PROCESSOR PROCESSING COMPLEX
AND METHODS", a copy of which is placed on the file. However, the
invention may be employed in other network processor-based devices of
various hardware/software designs.

5

Generally, as described in the aforementioned co-pending U.S. Patent Application, the general flow of a data frame received at a NP device is as follows. Frames received from an network connection, e.g., Ethernet MAC, are identified as either normal data frames or system control frames (Guided Frames). In the context of the preferred embodiment, frames identified as normal data frames are enqueued to an Embedded Processor Complex (EPC) which comprises a plurality of picoprocessors, e.g., protocol processors, which execute logic (picocode) capable of looking at the received frame header and deciding what to do with the frame (forward, modify, filter, etc.). The EPC has access to several lookup tables, and classification hardware assists to allow the picoprocessors to keep up with the high-bandwidth requirements of the Network Processor. A classification hardware assist device in particular, is provided for classifying frames of well known frame formats. The Embedded Processing Complex (EPC) particularly provides and controls the programmability of the NP device chip and includes, among other components (such as memory, dispatcher, interfaces), one or more processing units which concurrently execute picocode that is stored in a common instruction memory. Each processing unit preferably includes a Processing Unit core, for example, comprising a 3-stage pipeline, general purpose registers and an ALU. In operation, classification results from the classification hardware assist device are passed to a one or more processing unit during frame dispatch. One processing unit, in particular, referred to as a General Data Handler (GDH) comprises a full processing unit and one or more co-processing devices primarily used for forwarding frames according to a preferred embodiment of the invention.

The API of the preferred embodiment, in particular, is used by a GPP to configure parameters which affect where picocode executing at a particular network processor sends certain packet types. This API allows an end user to configure a customizable table that cross-references special frame types with a target GPP provisioned for handling the corresponding special frames. In this manner, the requirement of provisioning a protocol stack in each GPP is avoided.

Figure 3 depicts an example customizable table 100 for forwarding the special frame types to a pre-specified GPP via the API. In the preferred embodiment, the customizable table 100 is a flat table configured in NP device memory and includes entries corresponding to classified special frame types including, but not limited to: a Point-to-point (PPP) control type 102, unknown data frame types 105 (i.e., frames of a protocol not understood by the NP device), IP frame types 107,

6

and Wrap type 110 frames (i.e., frames received from a switch fabric at an egress side of the NP). It is understood that additional special frame types, e.g., layer 3 protocols, may be mapped for particular GPPs destinations in the customized table depicted in Figure 3. It is further understood that if the customized table 100 is of a larger size having many frame type entries, it may be configured in the form of a binary tree in the memory of the NP to facilitate table lookup. For each frame classified as a special frame type, there is a corresponding entry provided that includes: a target "blade" address 115, i.e., the address of the printed circuit board element associated with the destination GPP for handling the special frame type; and, a port address 120 of the NP device that connects to the destination GPP. It is understood that the table 100 in some embodiments includes additional mapping data as well.

In the preferred embodiment, the customizable table 100 is downloaded from the GPP to each network processor 125a,..,125n in the system 100 via the API. What follows is an example API implemented for configuring the flat table:

```
/*******************************************************
 * np_ims_customizedParms_configure()
 * INPUTS:
 * number_TLVs - number of TLVs that will be included
 * in the *tlv parameter
 * tlv - pointer to a string of Tagged List
 * Values (TLVs). Each TLV takes the
 * form:
 * +-----+-----+-----------
 * |Attr#|length|Value....
 * +-----+-----+-----------
 * Attr# - two octets attribute number
 * describing stored value.  See
 * np_types.h for possible
 * NP_ATTRNUM_'s.
 * length - two octets TLV length which
 * includes the length of the
 * Attr# and length fields.
 * Value - variable length field containing
 * stored value
 * ctrl_info - control information specifying the
 * expected asynch response characteristics
 * OUTPUTS:
 * none
 * RETURN VALUE:
 * np_return_code_t-
 * NP_RC_NOT_READY - IMS was not properly initialized
 * NP_RC_BADPARMS - Bad parameter passed
 * NP_RC_MISCERR - An error occurred during config
 * NP_RC_SUCCESS
 * ASYNCH RESPONSE:
 * This API will provide an asynchronous command completion
 * response as specified in the np_msg_ctrl_info_s param.
 * DESCRIPTION:
```

7

```
* This API is used to configure the customized parameters
* to direct picocode on where to send PPP control packets,
* IP packets, and unknown data packets (e.g., data
* protocols other than PPP, IP, etc.).
* The following TLVs are valid for this API:
* NP_ATTRNUM_PPPCONTROL_ADDR: 2 bytes NP + 2 bytes port
* NP_ATTRNUM_IP_ADDR: 2 bytes NP + 2 bytes port
* NP_ATTRNUM_UNKNOWN_DATA_ADDR: 2 bytes NP + 2 bytes port
* NP_ATTRNUM_WRAP_ADDR: 2 bytes NP + 2 bytes port
**************************************************/
np_return_code_t np_ims_customizedParms_configure(
        np_unit32number-TLVs,
        np_TLV_s*tlv,
        np_msg_ctrl_info_s ctrl_info);
```

The foregoing example API is now described in greater detail as
follows: Specifically, the API implements a procedure call
np_ims_customizedParms_configure, which in one embodiment is executed in
the C/C++ language, for receiving user-specified parameters for
configuring the customized table of Figure 3. The parameters include: for
example, a **np_unit32number-TLVs** input which represents the number of ·
tagged list values (TLVs) provided in a TLV list, and provides an
initialization of the number of special frame types that are to be
configured in the table; a **np_TLV_s*tlv** input which represents a pointer
to a string of TLV values; and, a **np_msg_ctrl_info_s ctrl_info** input which
represents control information for an asynchronous system response, such
as, a notification that the API has finished customizing the table for the
NP. With reference to the TLV value string, each string is preferably of
the above-identified form

```
+------+-----+-----------
|Attr#|length|Value....
+-----+-----+-----------
```

where Attr# represents an attribute number (one or more octets)
describing the particular protocol that is to be configured, e.g., PPP
protocol; length represents the physical length of the value; and Value
represents the actual value of the protocol frame type and comprises an
amount of bytes, e.g., two bytes,· representing the NP "blade" address and
another amount of bytes (e.g., two bytes) representing the NP port
associated with the target GPP. These values are downloaded for storage
in a variable length field.. A header file **np_types.h** may be referenced
for retrieving the possible number of TLVs and, in correspondence with the
customized table of Figure 3, include
NP_ATTRNUM_'s including two byte NP address + two byte port address such
as: NP_ATTRNUM_PPPCONTROL_ADDR, NP_ATTRNUM_IP_ADDR,
NP_ATTRNUM_UNKNOWN_DATA_ADDR, and NP_ATTRNUM_WRAP_ADDR.
Other data configured for download via the API include control information
for specifying any expected asynchronous response characteristics, such

8

as, for example, a notification via the API that each of the NPs in the
network have been configured with the customized table. A return
parameter of the API referred to as **np_return_code_t**, for example, returns
values such as: NP_RC_NOT_READY for indicating that a particular
sub-system was not properly initialized; NP_RC_BADPARMS for indicating
that a bad parameter was passed; NP_RC_MISCERR for indicating that an
error occurred during table configuration; and, NP_RC_SUCCESS for
indicating that the NP configuration was successful.

Having configured the EPC of each network processor in the
distributed network, the process 200 of distributing processing across the
several general purpose processors is now described with reference to
Figure 4 and in accordance with a preferred embodiment.

As shown in Figure 4, there is depicted a first processing step 203
which depicts a network processor receiving a frame from the network. At
step 205, the received frame type is classified by a hardware classifier
of the network processor device and the frame is forwarded to the NP's
picocode executing in the GDH. If the hardware classifier was not able to
classify the frame, the picocode classifies it.   Based on the
classification, at step 210, the picocode executes logic for performing a
lookup into the customized table 100 (Figure 3) loaded in the data store
of a network processor. It is understood that the frame type is
preferably used as an index into the table 100. Then, at step 213, a
determination is made as to whether a match exists, i.e., whether the
frame is to be forwarded to a specific GPP for special processing based on
its frame type. If no match exists (i.e., no special processing is
required), the network processor processes the frame at step 215. If, at
step 213, it is determined that a match exists, the lookup is performed in
the customized table 100 to determine the target GPP address and port
address for that frame type at step 218 and, the special frame is
forwarded to the appropriate GPP at step 220 using those parameters. The
processing described with respect to Figure 4 results in four possible
outcomes: 1) the packet is forwarded for processing in the network
processor; 2) the packet is redirected to a GPP for processing; 3) the
packet is dropped; and 4) the packet is dropped and an unsolicited message
is sent to the GPP.

An example using the methodology of the preferred embodiment
depicted in Figure 4 is now described for the case involving processing of
a Point-to-Point Protocol (PPP) frame. The PPP frame is processed by the
network processor. If the frame is a PPP control frame, it is forwarded

to the configured GPP based on the customized lookup table (Figure 3). If
the frame is a Layer 3 frame, the Layer 2 network processor picocode
forwards it to the appropriate network processor component. If the Layer
3 protocol is not handled by picocode, the frame is forwarded to the
configured GPP for that protocol as specified in the lookup table. If the
protocol has not been enabled, the frame is enqueued to a discard queue,
not shown. If the frame is not a recognized Layer 3 frame and the discard
unknown PPP frame is configured, it is enqueued to the discard queue.
Otherwise, the frame is forwarded to the configured GPP for
unknown/unrecognized PPP frames as specified in the lookup table.

Exemplary actions taken are summarized in the following Table:

Protocol	Action
PPP Control frame	Redirect to configured GPP
IP	Proceed to NP IP Layer 3 if IP enabled in PCT
Unknown	Redirect to configured GPP (Unknown Data) if the unknown PPP frame is not set in the PCT.

It is understood that by enabling a network processor to distribute
processing across several general purpose processors, and providing the
capability of configuring which NP handles specific operations in a large
networking environment, advantageously results in the requirement of fewer
protocol stacks on each general purpose processor, as each GPP will be
equipped to handle only those special frames forwarded by the NP in
accordance with the customizable table exemplified in Figure 3.

CLAIMS

1.   A system for distributing processing of special data frames received
by a network processor (NP) device (25a, 25b, 25c...25n) in a distributed
networking environment comprising one or more general purpose control
processors (GPP) (15a, 15b...15n) that control at least one NP devices,
said system comprising a configurable table (100) implemented in said at
least one NP device for mapping special data frame types received at said
NP device with a target address (115) and a target port address (120)
associated with a target GPP capable of handling said special frame type,
wherein said NP network processor includes mechanism for classifying the
received data frames and comparing each said frame to entries in said
configurable table, and said network processing device includes means
responsive to said classified frame type having a corresponding entry in
said table, for forwarding said frame to said target GPP through a
corresponding target port address, wherein said target GPP is provisioned
for handling forwarding of said special data frame type.

2.   The system according to Claim 1, further including an application
programming interface for customizing said configurable table in said NP
device with said target address and target port address entries through
said GPP.

3.   The system according to Claim 1 or 2, wherein a special data frames
type includes a data frame of a protocol unrecognized by said NP device.

4.   The system of Claim 1 or 2 comprising:

      configurable means for enqueuing frame types of a protocol
unrecognised by said NP device to a discard queue.

5.   The system according to Claim 1, 2, 3 or 4, wherein said
configurable table is of a flat table design configured in memory of said
NP device.

6.   The system according to Claim 1, 2 , 3 or 4, wherein said
configurable table is of a binary tree design configured in memory of said
NP device.

7.   A method for distributing processing of special data frames received
by a network processor (NP) device (25a, 25b, 25c...25n) in a distributed
networking environment comprising one or more general purpose control

processors (GPP) (15a, 15b...15n) that control one or more NP devices,
said method comprising:

    a) providing a table (100) implemented in said NP device for mapping
special data frame types received at said NP device with a target address
(115) and a target port address (120) associated with a target GPP capable
of processing said special frame type;

    b) receiving at said NP network processor a data frame of a data
frame type (203);

    c) classifying the received data frames and comparing each said
frame to entries in said table (205); and

    d) if said classified frame type has a corresponding entry in said
table, forwarding said special data frame to said target GPP through a
corresponding target port address, said target GPP provisioned for
handling forwarding of said special data frame type (220).

8.   The method of Claim 7, comprising the step of providing a
configurable option to enqueue to a discard queue frame types of a
protocol unrecognised by said NP device.

9.   The method for distributing processing of special data frames
according to Claim 7 or 8, wherein step a) includes the step of:
implementing an application programming interface for configuring said
table with said target address and a target port address associated with a
target GPP for each special frame type entry.

10.   An application programming interface for configuring a network
processor (NP) device (25a, 25b, 25c...25n) operating in a distributed
networking environment including one or more general purpose control
processors (GPP) (15a, 15b...15n) that control one or more NP devices,
said interface comprising:

    a mechanism for generating a data structure for input to said NP
device, said data structure having one or more entries specifying a
special frame type and associated target address (115) and target port
address (120) associated with a GPP in said networking environment capable
of handling said special frame type; and,

a device for generating a customized table (100) from said generated data structure, and inputting said customized table to an NP device memory, whereby subsequent table lookups in said table are performed for forwarding special frame types to an associated GPP through a target port address.

11.   The application programming interface according to Claim 10, wherein said data structure includes a string of tagged list values, each tagged list value including an attribute field representing a special frame type to be included in said table, a length field representing a length of said tagged list value, and, a value field for specifying a target associated target address and target port address for said special frame type.

12.   A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for configuring a network processor (NP) device (25a, 25b, 25c...25n) operating in a distributed networking environment including one or more general purpose control processors (GPP) (15a, 15b...15n) that control one or more NP devices, said method steps comprising:

    (a) generating a data structure for input to said NP device, said data structure having one or more entries specifying a special frame type and associated target address (115) and target port address (120) associated with a GPP associated with said networking environment capable of handling said special frame type; and,

    (b) generating a customized table (100) from said generated data structure, and inputting said customized table to NP device memory, whereby subsequent table lookups in said table are performed for forwarding special frame types to an associated GPP through a target port address.

13.   The program storage device readable by a machine as claimed in Claim 12, wherein said data structure includes a string of tagged list values, each tagged list value including an attribute field representing a special frame type to be included in said table, a length field representing a length of said tagged list value, and, a value field for specifying a target associated target address and target port address for said special frame type.
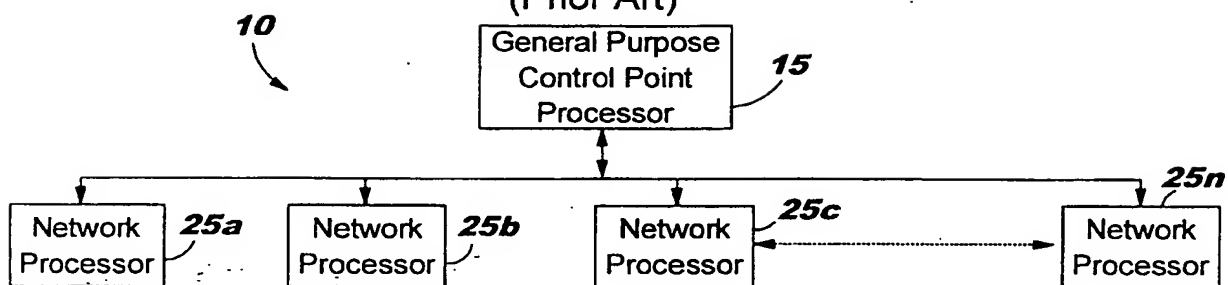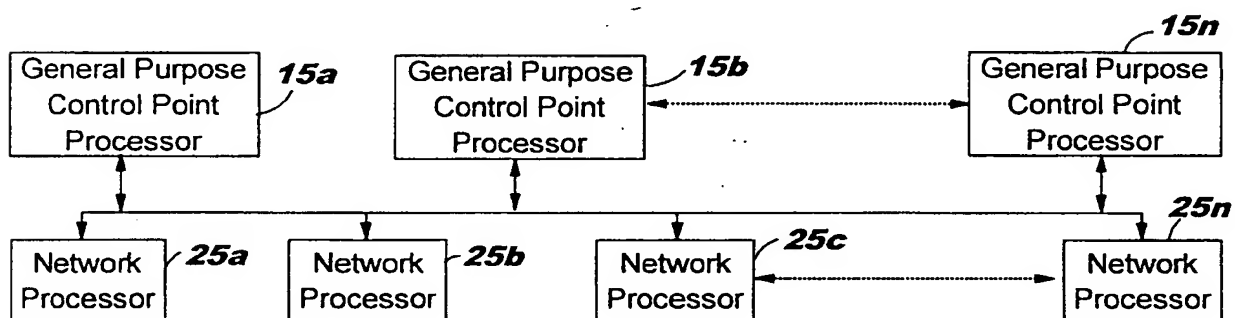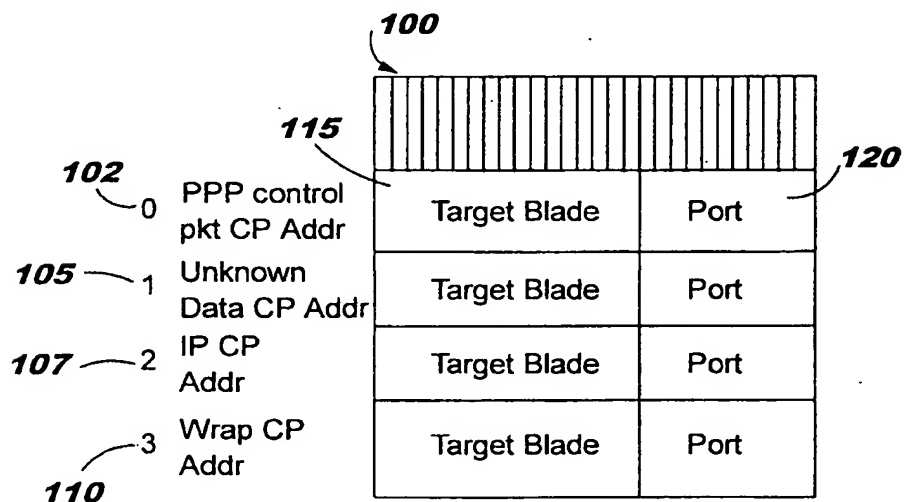
FIG. 1
(Prior Art)

FIG. 2

FIG. 3

FIG. 4

200

```
        ┌─────────────────────┐   203
        │  Network Processor  │
        │   Receives Frame    │
        │    From Network     │
        └─────────────────────┘
                   │
                   ▼
        ┌─────────────────────┐   205
        │     Frame Type      │
        │     Classified      │
        └─────────────────────┘
                   │
                   ▼
        ┌─────────────────────┐   210
        │  Frame Type is One  │
        │  Found Compared to in │
        │   Customized Table  │
        └─────────────────────┘
                   │
                   ▼
             ◇ 213
           Does Frame          Yes      ┌─────────────────────┐  218
           Need Special      ────────▶  │ Lookup parameters for │
           Processing?                  │    Frame Type in    │
             ◇                          │   Customized Table  │
                   │                    └─────────────────────┘
                   │ No                            │
                   ▼                               ▼
  215    ┌─────────────────────┐         ┌─────────────────────┐  220
         │   Frame Processed   │         │    Send Frame to    │
         │     by Network      │         │     Destination     │
         │      Processor      │         │  Identified During  │
         └─────────────────────┘         │    Table Lookup     │
                                         └─────────────────────┘
```